Sentiment analysis of student reviews of student writing

Michael Wojcik
Rhetoric & Writing
Michigan State University
East Lansing, Michigan
wojcikm4@msu.edu

Abstract

Eli is a web-based application for coordinating and assessing writing review, typically used by students in English composition classes at the secondary or undergraduate level. It provides students and instructors with information about the reviews that students write of one another's writing, based on reception (whether the original author finds the review helpful), but does not currently attempt to compute any metrics based on the content of the reviews themselves. I examine some sentiment-determination approaches, implemented using the UIMA framework, and their applicability to the Eli review data. The long-term goal of this project is to use sentiment-determination techniques to analyze the rhetorical tone—the attitude of the writer toward the material—of reviews. I describe how this information may be useful for instructors, and how the introduction of NLP techniques into an application like Eli can extend its capabilities; and discuss what sentiment analysis might tell us about writing review.

1 Credits

This project was developed for Joyce Chai's CSE841 Natural Language Processing class at Michigan State University, Spring 2011. The original Eli research project was developed by Mike Mcleod and Bill Hart-Davidson at the WIDE (Writing in Digital Environments) Research Center at Michigan State University. WIDE subsequently entered a partnership with Red Cedar Solutions Group in Okemos, Michigan, to produce a commercial version of Eli, which is currently in beta-test stage.

2 Introduction

Eli, a web-based application for coordinating and assessing writing review, is an innovative approach to teaching (writing) composition, particularly at the secondary and undergraduate levels, and potentially a useful tool for professional writing management as well. It grew out of work at the Writing in Digital Environments (WIDE) Research Center at Michigan State University which was aimed at applying the tools of "social media" websites to contemporary composition pedagogy, and is now being developed into a commercial product for the education market (WIDE, 2011b). Eli provides writers, typically students in writing classes, with an environment in which they can read and review one another's writing, and then provide feedback on the reviews they receive in turn, so they can become better reviewers as well as better writers. Research in language-composition pedagogy has shown that peer review coupled to a process for improving reviewing skills is one of the most effective ways to improve a writer's performance; Eli is designed to streamline this process.

In the Eli environment, students can post their own writing, then read other students' texts and write free-form reviews of them. Reviewers also rate each document on a 1-5 scale; the precise meaning of this rating is deliberately left vague, but a "1" indicates a document that requires the most revision, and a "5" one that is very good in its current form. These numerical ratings are useful in the project,

as I explain below. (Reviewers may also respond to more structured review questions set by the instructor, such as Likert-scale ratings and checklists of requirements. Instructors can provide "prompts" to guide students in writing their reviews.) As students receive reviews on their own writing, they can evaluate those reviews, and develop plans for revising their own writing. Meanwhile, Eli shows them how the reviews they created were received by their fellow authors. So each Eli user does four kinds of writing: primary text, reviews, responses to reviews, and revision plans. Part of Eli's function is simply to provide space for and coordinate all of these activities. Beyond that, though, it tracks user actions such as creating and viewing objects (primary texts, reviews, etc), highlighting or copying text, commenting, and so on.

Eli calculates a *helpfulness metric* (now the subject of a patent application by the university) which gauges the utility of a review based on the receiving author's actions and feedback (Hart-Davidson et al., 2010). For example, when an author copies a suggestion from a review onto the revision plan for the next revision of the writing assignment, Eli notes that action and increases the reviewer's helpfulness score. Eli maintains a cumulative helpfulness score for each user (student), based on the average helpfulness of that user's reviews. Instructors use this information to track student performance as a peer reviewer, and students use it to improve their reviewing skills. However, Eli currently does not attempt to compute any metrics based on the actual *content* of reviews, only what the reviewed author does with them.

This is an application that appears ripe for the application of Natural Language Processing techniques, to give the Eli system mechanisms to estimate additional helpfulness characteristics and other information about the reviews that users write. In fact, WIDE has been discussing this possibility since at least 2008, when I begin some exploratory work for the Center on heuristics for gauging an author's *ethos*, or reputation among members of the audience (Wojcik, 2009). In this project I review the UIMA data-processing framework as a possible approach for adding NLP features to Eli, and describe three prototype modules I implemented using UIMA (and other open-source components) to do sentiment analysis on student reviews. I evaluate my selected sentiment-determination algorithms against my manual determination of sentiment for the reviews in the data set. After discussing what sentiment analysis might tell us about writing review in theory, and how the sentiment-analysis information might be used by instructors, I touch on some of the broader implications of what NLP might do for Eli.

WIDE has agreed that this project falls under the scope of their existing Eli research, which the IRB has determined does not require human-subjects research approval.[1]

3 Sentiment Analysis

The literature on sentiment analysis (including sentiment classification, extraction, retrieval, aspect identification, etc) is large. This is a popular area of NLP research, no doubt in part because of its commercial value: it is used by media-analysis professionals in government, public relations, and similar fields; by market analysts studying online product reviews; and in other well-funded areas. But it is also of considerable theoretical interest, since it incorporates syntactic, semantic, and pragmatic problems in language analysis; often deals with unstructured and noisy data;[2] and seeks to extract high-level, often implicit meaning from the input.

Liu (Liu, 2009) is a useful overview of the problem of sentiment analysis and common formulations of this kind of work, including a brief look at the application to online product reviews, one of the most frequently discussed areas.

Many older approaches to sentiment analysis used a bag-of-words model that tried to identify sentiment-bearing terms. Gamon and Aue (Gamon and Aue, 2005) describe an automatic approach along these lines. As Nakagawa *et al.* (Nakagawa et al., 2010) point out, however, bag-of-words models inevitably have a high error rate because sentiment terms are often embedded in phrases that invert their meaning (e.g. "it's not that I dislike"). They present an approach using hidden variables representing the sentiment of dependency subtrees, which are com-

[1] Bill Hart-Davidson is the primary investigator for the original Eli project.

[2] Online reviews, for example, are notorious for misspellings and other errors, slang, and inconsistencies. Brody and Elhadad (2010), among others, discuss some of these problems.

bined to infer the sentiment of the sentence. This is much more complex and computationally expensive than a bag-of-words approach but has significantly better accuracy.

Arora *et al.* (Arora et al., 2010) go further, with a model that combines bag-of-words features and structural features, the latter derived from linguistic annotation graphs; they use a genetic-programming approach to reduce the large number of features extracted by their model to a salient subset. While this approach looks very powerful, it is also likely to be resource-intensive both in implementation and execution. Another novel approach, called HL-SOT for "Hierarchical Learning with Sentiment Ontology Tree," is described by Wei and Gulla (Wei and Gulla, 2010). They focus on solving two problems with sentiment analysis of product reviews in particular (though their method seems applicable to other problem domains): applying information contained in the hierarchy of attributes of the topic being reviewed, and coping well with sentences that express complex sentiments about multiple attributes. As with the system presented by Arora *et al.*, I found Wei and Gulla's method interesting but too complex to implement for this project.

Greene and Resnik (Greene and Resnik, 2009) move away from lexical approaches like bag-of-words to focus on syntactic "packaging," or how key ideas are arranged in the phrasal structure of the input. They present a strong argument for the presence of implicit sentiment cues for readers in syntactic structures, and their method has some success at identifying these. Another recent proposal, from Brody and Elhadad (Brody and Elhadad, 2010), has two especially interesting features. One is that it incorporates information from topic aspects, which the authors claim often dominate sentiment interpretation: for example, "cheap" may be positive when referring to restaurant prices, but negative when applied to restaurant decor. The other is that their system is unsupervised, which not only means it's cheaper to implement, but, according to the authors, deals well with irregular data (unrecognized words, errors, etc).

3.1 Sentiment Analysis of Writing Reviews

Because student-writing reviews are a more focused genre than many of the domains where sentiment determination is currently popular (for example, product comments forums for online shopping sites), and since the writers feel some constraint to produce well-formed text, the input has relatively low noise, but sentiment markers also tend to be more subtle. Students are generally reluctant to express overtly negative opinions of one another's writing, particularly when author and reviewer are part of a small group such as a class.[3] They may try to couch their negative opinions "constructively" or in an ostensibly positive gloss, so e.g. "your second paragraph doesn't make sense" might become "I think your second paragraph might be better if you explained your ideas more". They also often preface their substantive remarks with formal expressions of approval ("overall I liked your essay"), as a sort of social nicety; this will distort a straightforward sentiment evaluation. One more impediment to straightforward evaluation is that student reviewers will often express sentiment about the topic of the piece they're reviewing, which should not be counted as a sentiment about the writing itself.

For those reasons, I hypothesized that a bag-of-words approach would not be very successful. Nonetheless I implemented one, as a baseline, but expected more sophisticated methods to significantly outperform it. However, in my testing (which should still be considered preliminary) my bag-of-words analyzer tended to give the best results of my candidate analyzers. I suggest below that this is largely due not to the superiority of a bag-of-words approach, however, but to issues with training, tuning, and augmenting the others. My candidates for improved methods are a higher-order n-gram analyzer that uses trigram and bigram data in addition to unigrams, with various tuning parameters; and a phrase-based analyzer that makes some use of syntactic structure, and also tries to detect *polarity reversal* (when the "base" sentiment of a term is inverted by context).

[3] Eli can be configured to anonymize authors and reviewers, and it can be used to have users review the work of authors they have no other connections with; but typically reviewers and authors are all members of a group, and are aware of that connection.

4 UIMA as an NLP Framework

Because part of my goal is to open Eli up to future NLP-based projects, I intend to implement a general framework for NLP of the data contained in Eli as part of the larger project. To that end I developed my sentiment analysis modules within the Apache implementation of the Unstructured Information Management Architecture (UIMA) framework as the overall structure, and used existing Apache and OpenNLP UIMA modules for basic tagging and parsing. UIMA is a standard framework for handling unstructured data, particularly the sort of noisy *ad hoc* text found on social-networking websites and the like, originally developed at IBM and now standardized by OASIS (Apache Foundation, 2011). It's used in a wide range of applications, recently and famously in the IBM Watson Jeopardy-winning system (Pearson, 2011). IBM handed the code base off to the Apache Foundation for maintenance; while anyone can implement UIMA, in practice Apache UIMA is the implementation most people use. Apache UIMA is available in source and compiled form and comes with a number of useful sample modules, plus documentation, tutorials, and the like. It is designed primarily for use with C++ and Java, and includes enhancements to the Eclipse IDE for those languages, but it supports other programming languages through bridge interfaces. Programs written using Apache UIMA can be packaged as modules for use in larger projects, as standalone applications, as web services, or as worker roles for distributed systems such as Hadoop or Apache's own UIMA-AS (UIMA Asynchronous Scaleout). UIMA itself consists of a modular processing architecture and a data schema and type system for representing subjects, analyses, annotations, and similar objects of interest. Besides the Apache samples, there are a number of free, open-source UIMA modules that a developer can make use of, including wrappers for a number of OpenNLP functions.

UIMA is useful for analytic NLP tasks in a number of ways. It's easy to represent common NLP functions such as part-of-speech tagging as UIMA *annotators*, which can attach arbitrary metadata to spans of input data in a fashion that's easy for downstream modules to consume. The modular architecture of UIMA makes it easy to build processing pipelines of these basic NLP functions, freeing experimenters up to focus on problems of specific interest. Because UIMA is standardized, popular, and free, it's convenient for sharing work with other researchers; its support for loose coupling (*e.g.* deploying applications as web services) makes it easy to provide other researchers with a sandbox in which to try out a new feature. Its scale-out capabilities help with processing large corpora of data, and with very compute-intensive research problems.

For this experiment, I created a number of UIMA annotators and *descriptors* (XML files used by UIMA to describe types, annotator workflows, and other aspects of runtime configuration) to experiment with sentiment analysis of review data extracted from Eli. I ran these under the UIMA Document Analyzer application, a convenient utility for UIMA experimentation. In the longer term, if we find this kind of work is useful for Eli, we will more likely expose it as a web service and integrate it loosely into the Eli system. That will permit independent development on Eli and these NLP features for it. We would likely also support both online and batch-mode execution, so that Eli users could instantly get results for a select set of reviews, or process a larger corpus of them in the background.

5 Sentiment Analysis and Review Tone in Eli

The original impetus of this project was to attempt to computationally discern *tone* in the student review documents. Rhetorical tone is a more abstract and complex concept than simple sentiment. Part of my larger project, which falls into the area I call *computational rhetoric* (Wojcik, 2011), is to investigate to what extent we can heuristically determine rhetorical tone based on extracted features such as sentiment. At this stage, however, I am simply focusing on sentiment under the assumption that it can be used as a proxy for tone.

I believe this information has at least three possible uses. First, instructors may find that students respond differently toward reviews based on their tone, and use that information to make suggestions to students about how to couch their review comments. Second, review-tone information might be used to flag certain problematic reviews for instruc-

tor attention. Third, tone analysis of reviews could begin to address open research questions such as whether the helpfulness of a review is correlated to its tone or perceived sentiment, or whether there is a correlation between sentiment and the numerical rating the reviewer assigns to the original text.

6 Data and Representation

The data for the study was extracted from a snapshot of the Eli database that was generated during beta testing. For ease of processing, I unloaded the relevant columns from the database to text files, rather than operating on data in the database. Each file contained the database unique row ID for a review (for identification purposes; this ID was also used as the file name), the numerical rating assigned to the original text by the reviewer, and the text of the review. The extraction process skipped rows that had no review text and some test rows that contained dummy reviews (*lorem ipsum* text). It also removed HTML tags from the review text and converted some HTML entities and non-ASCII characters to their ASCII equivalents. This preprocessing was done with an *ad hoc* script written in GNU Awk.

After extraction and filtering, there were 3659 review documents. They have a wide range of length (from a few words to multiple paragraphs), writing style, and content. Some are simply factual information ("could not open document") or short subjective evaluations ("too technical?"). Others, however, express relatively complex critiques with substantial sentiment markers. There is some use of slang and abbreviations, and the occasional misspelling, but in the main the writing is relatively standard and amenable to processing.

The numerical rating that accompanies each review is an overall quantitative evaluation of the piece being reviewed, in the "four of five stars" mode. In theory, numerical ratings range from 1 (lowest) to 5, but in fact none of the reviewers in this database snapshot had ever assigned a "5" rating. There are likely various reasons for this; for example, students participating in the review process feel charged to provide helpful commentary, which implies that the piece they're reading has room for improvement, and so can't be given the highest possible rating. These numerical ratings are of interest for various reasons. On one hand, we would like to discover whether there's any correlation between numerical rating and computed sentiment—does negative sentiment always mean a low rating? On another, if we assume there is such a correlation, we can use ratings to assign reviews to negative-sentiment and positive-sentiment buckets for training purposes. In fact, this technique was used, as described below.

After extraction, the set of documents was partitioned in various ways for further analysis. First the documents were grouped according to numerical rating. Then 100 each of rating-1 and rating-4 documents were set aside for training purposes. A subset of 10 documents (5 rating-1 and 5 rating-4) were copied from the training set to test how well the sentiment analyzers performed against data they had been trained on. I evaluated sentiment manually in these 10 so that we were not relying simply on the numerical ratings (which did not contain sufficient information anyway, as discussed below). I also took another 40 documents, 10 of each rating, and manually evaluated sentiment in them; these 40 plus the previous 10 became the gold-standard set used to evaluate analyzer performance.

6.1 Sentiment Representation

Early sentiment analysis work often simply represented sentiment as either "positive" or "negative". Some later approaches, such as that of Gamon and Aue (Gamon and Aue, 2005), added a "neutral" category for documents that don't appear to include any significant sentiment. Other projects have used more sophisticated representations for sentiment, including the "strongly" and "weakly" positive and negative tags in the MPQA corpus (Wiebe, 2002).

For this project, I defined positive and negative sentiment as independent binary features: a document could have no significant sentiment (aka "neutral"), primarily positive sentiment, primarily negative sentiment, or both positive and negative sentiment ("mixed"). In addition to these textual labels I defined this as a two-bit integer, with the positive-sentiment feature assigned to the more-significant bit, so a numeric value of 0 sentiment indicates neutral, 1 indicates predominantly negative, 2 positive, and 3 mixed.

For purposes of computing precision and recall when evaluating sentiment-analysis implemen-

tations, I treat neutral as the lack of any features, and calculate separate results for positive-sentiment alone, negative-sentiment alone, and the two combined.

7 Sentiment Analysis Implementations

I use three algorithms for analyzing sentiment in the Eli reviews, all implemented as UIMA annotators, specifically as the final primitive annotator in a sequence of annotators that also includes basic NLP functions such as tokenizing and sentence boundary detection. I experimented with a fourth algorithm as a standalone program but did not develop it into an annotator because I deemed not worth the necessary additional effort at this stage.

7.1 Bag-of-Words Analyzer

The first algorithm is a relatively straightforward machine-trained bag-of-words analyzer. It compares each word in the review text with predefined lists of terms that it associated with positive-sentiment and negative-sentiment reviews during the training stage. For each match, it incrementes the appropriate total-sentiment variable by the MLE probability that the found word belongs to the positive or negative class. Finally, it determines overall sentiment by comparing the positive and negative totals.

Training is done by a separate program. The training algorithm extracts all the unique words from the 200 training documents and their occurrences in each of the two groups of documents. Words with a count less than 7 were discarded. Then the lists were compared, and words that appeared on both were either removed from the list where they had a lower MLE probability, or from both lists, depending on whether the greater probability exceeded the lower by at least a factor of 1.5. (This value was selected arbitrarily but appears to give usable results.) The resulting lists are disjoint, obviously, and represent terms likely to appear in either a positive or a negative review, based on the training data.

It's worth noting that these lists do not in general correspond to any literal expression of sentiment. For example, terms in the negative list include "outline", "start", and "other"; the positive list has "argument" and "maybe". These are simply terms that students writing the reviews used in training tend to use when creating comments that expressed negative or positive sentiment.

In order to distinguish significant sentiment from neutral documents that simply contain a few terms from the lists, the bag-of-words analyzer applies a threshold value; calculated sentiment must exceed the threshold in order to appear in the results. The analyzer also requires that one of the sentiment features (positive or negative) exceed the other by a predefined margin, or it evaluates the document as having mixed sentiment. Currently, the threshold value is 3 sentiment-related terms, and the margin is a factor of 1.5; these are also arbitrary values I selected intuitively which appear to give reasonably useful results.

7.1.1 Sentence-Mode Analysis

The bag-of-words analyzer has one other feature, which actually does add some consideration of linguistic structure: *sentence-mode* analysis. When this option is enabled, the analyzer processes the document sentence by sentence, rather than simply word by word. Within each sentence, sentiment-associated words are recognized, as per normal. However, at the end of each sentence, determination is made about the sentiment of that sentence based on the words found and the threshold and margin values. After all sentences have been processed, the document's sentiment is based on the relative counts of positive and negative sentences. (At this point the threshold is zero, so the document will not be considered neutral unless no sentiment-bearing sentences were identified.)

I introduced sentence-mode analysis on the intuition that in this genre, authors generally try to confine each sentence to one idea or a few related ones, and so sentiment will generally be consistent in a sentence. Thus sentence-level granularity of sentiment detection helps reduce noise. In practice, sentence-mode analysis did give slightly better results in testing, as noted below.

7.2 Trigram Analyzer

The second analysis algorithm attempts to incorporate some local context and provide better identification of sentiment-bearing phrases by incorporating bigrams and trigrams into the recognized-terms lists. These n-grams are learned from the training sets of

positive- and negative-sentiment reviews; then the analyzer scans input review texts for matching n-grams. This algorithm also uses the MLE probability of an n-gram to weigh the n-gram's contribution to the current total positive or negative sentiment.

The current implementation of the trigram analyzer offers a couple of simplistic modes for combining the probabilities of n-grams of different order, neither of which gives very good results in the testing done to date. The default mode is interpolation, where the λ_n coefficients are either assigned by the user or taken from a default set picked by trial and error (as time did not permit discovering optimal values using a held-out corpus and an optimization procedure). The alternative is "false backoff", where the algorithm simply backs off to the probability of successively lower-order n-grams when the higher-order ones have zero probability. Of course this is not a valid probability function, but I implemented it largely to see what results it would produce. In testing it performed worse than interpolation, because lower-order n-gram MLE probabilities are greater than or equal to the corresponding higher-order ones, and so sentiment-relevant expressions that matched higher-order n-grams were penalized in the sentiment-counting phase.

Like the bag-of-words analyzer, the trigram analyzer offers a sentence-oriented mode. It also has a number of other configuration parameters, such as a *sensitivity* setting that affects the threshold and margin values (which serve the same role as they did for the bag-of-words analyzer) and interpolation coefficients.

While the language model for the bag-of-words analyzer was trained using a separate utility, the UIMA implementation of the trigram analyzer includes its own training mode; training or analysis is selected by which UIMA descriptor is used. This demonstrates some of the flexibility of the UIMA architecture.

7.3 Phrasal Analyzer

The third algorithm tries to make more sophisticated use of linguistic structure, an important issue for sentiment estimation—the bag-of-words model often results in false-positive results because it finds a term that is often sentiment-bearing, but in a context that removes the sentiment, inverts its polarity, or applies to something other than the presumed subject of the text. The local context used by the trigram model is only good for short phrases (*e.g.* "not very good" as a negative, rather than positive, sentiment expression); human readers can detect subtleties of sentiment that are expressed by complex components of sentence structure. So this phrasal, or phrase-based, analyzer always operates at sentence-level granularity, and attempts to make use of phrase structure to detect polarity reversal.

The phrasal analyzer is also of interest to researchers working with open-source NLP tools because it integrates UIMA with components of the OpenNLP project. Its UIMA application uses a tokenizer, sentence delimiter, part-of-speech tagger, and sentence parser from OpenNLP (and their associated data types), plus a token-filtering module and the actual sentiment analysis module written by me. As such, it's a useful example of how these disparate NLP technologies can easily be combined using UIMA.

This analyzer also differs from the others in that, rather than using a machine-learning approach, it constructs its language model from dictionaries compiled by other NLP projects. From the MPQA project (Wiebe, 2002) it gets a dictionary of sentiment-bearing words; these are labeled as strongly or weakly expressing sentiment, so the analyzer records that information too and uses it as a weight. From the General Inquirer project (GI,) it gets a list of words that negate the sense of a phrase, and another of ones that decrease the sense; these are used to detect polarity reversal, as explained below. (Sense-decreasing words are considered polarity reversers because of their use in phrases like "low quality", which is a negative-sentiment expression.)

The phrasal analyzer iterates over sentence parse trees, searching the leaves (representing individual words or phrases, as the OpenNLP parser is moderately shallow) for sentiment-bearing terms. If it finds any, it checks the immediate parent of that leaf—which will typically be a noun phrase or similar construct—for a reversal term. Then, if it has a result (positive or negative, strong or weak, possibly reversed from the simple sense of the word) for that leaf, it adds it to the total sentiment for the sentence.

This design was inspired and strongly influenced by the work of Nakagawa *et al.* (Nakagawa et al.,

2010), though it is far less sophisticated. Besides insights about phrase-level sentiment analysis, I took from them the suggestion to use the word lists from the MPQA and GI projects.

This is still a simplistic language model, but it does have the opportunity to consider some less-localized context, and specifically sentiment polarity reversal, by associating each sentiment-bearing term with the phrase it appears in. It also makes use of human-generated dictionaries, which may prove more accurate (current results are indeterminate) than the simple models generated from the relatively small amount of Eli training data.

7.4 Experimental Bigram Hidden Markov Model Analyzer

I experimented briefly with a bigram-based HMM analyzer, adapted from a conventional POS tagger where *sentiment tags* replace the part-of-speech tags. Here a sentiment tag is one of five values associated with each word: strongly negative, weakly negative, neutral, weakly positive, or strongly positive. The tag does not necessarily refer to the word itself, but to the change in sentiment since the last sentiment-bearing expression—for example, in a phrase like "your introduction is not as good as it might be", the word "might" could be tagged weakly negative, whereas the word "should" in the same position might merit a strongly-negative tag.

Unfortunately, getting decent results from this approach requires training it with a relatively large amount (on the order of thousands of sentences) of data which has been manually-annotated with sentiment tags by a human judge. This proved infeasible during the course of this stage of the project. It's possible that for another stage we'll develop a protocol for making this work more convenient and distributing it among a number of human judges.

8 Results

At this point in the project, the results can be summed up as sufficiently interesting to warrant further investigation, but not yet useful for direct research application. The bag-of-words analyzer, which was originally intended as a baseline to compare more sophisticated algorithms against, currently offers the best results in some categories, and is only bested slightly by certain configurations of the trigram analyzer in others. And even those are not terribly good. However, these are still very preliminary given the small set of gold-standard data (N=50) I have to compare algorithm output against.

	P	R	F_1	$F_{0.5}$
bag-of-words	0.48	0.48	0.48	0.48
BOW sentence	0.55	0.55	0.55	0.55
trigram	0.44	0.40	0.42	0.43
trig. sensitive	0.40	0.43	0.41	0.41
trig. sentence	**0.62**	**0.57**	**0.59**	**0.61**
trig. sensitive sentence	0.48	0.50	0.49	0.48
phrasal	0.49	0.48	0.48	0.49
phrasal no reversal	0.44	0.43	0.43	0.44

Table 1: Sentiment Algorithm Results—Joint Sentiment

8.1 Algorithm Accuracy

I provide results for the three UIMA-based analyzers, including two configurations each of the bag-of-words and phrasal analyzers, and four of the trigram analyzer. In addition to precision, recall, and F_1 value, I provide the $F_{0.5}$ value, which treats precision as twice as important as recall. The research we intend to do with sentiment-analysis results on Eli review data are exploratory—we are more interested in finding a consistent subset of the reviews with a certain sentiment than we are with finding all the reviews that likely have that sentiment. So precision is considered more useful for our purposes. I provide figures for precision and recall both taken jointly (in which case the sentiment determined by the algorithm must match the one I assigned manually), and take independently; in the latter case, when considering positive sentiment, a result of "positive" from the analyzer is considered a match if the gold standard value is "mixed", for example.

8.1.1 Bag-of-Words

In its default configuration, the bag-of-words analyzer produced F_1 values between 0.48 and 0.76, depending on whether we're considering joint or independent positive/negative features.[4] When run in sentence mode, the F_1 value for negative sentiment

[4] In a previous presentation of this material, I discovered I had incorrectly calculated the F_1 values for joint sentiment.

	P	R	F_1	$F_{0.5}$
bag-of-words	0.75	0.78	0.76	0.76
BOW sentence	**0.81**	0.78	**0.79**	**0.80**
trigram	0.70	**0.85**	0.77	0.72
trig. sensitive	0.68	**0.85**	0.75	0.71
trig. sentence	0.71	0.81	0.76	0.73
trig. sensitive sentence	0.69	0.74	0.71	0.70
phrasal	0.71	0.81	0.76	0.73
phrasal no reversal	0.71	0.81	0.76	0.73

Table 2: Sentiment Algorithm Results—Positive Sentiment Only

	P	R	F_1	$F_{0.5}$
bag-of-words	0.68	0.71	**0.69**	**0.69**
BOW sentence	**0.70**	**0.58**	0.64	0.67
trigram	0.50	0.17	0.25	0.36
trig. sensitive	0.56	0.38	0.45	0.51
trig. sentence	0.56	0.21	0.30	0.42
trig. sensitive sentence	0.67	0.42	0.51	0.60
phrasal	0.69	0.46	0.55	0.63
phrasal no reversal	0.60	0.38	0.46	0.54

Table 3: Sentiment Algorithm Results—Negative Sentiment Only

is somewhat lower, but for positive and joint sentiment that configuration returns better values. The bag-of-words analyzer was the best overall at determining independent positive and negative sentiment.

8.1.2 Trigram

I tested the trigram analyzer in a number of configurations. No one configuration excelled at all tasks. All four used interpolation with the default coefficients. I ran the analyzer with the default sensitivity in normal and sentence mode, and again with the sensitivity set close to the maximum (which means a very low threshold for deciding a sentiment marker is significant, and a very low margin for distinguishing between mixed sentiment and predominantly positive or negative).

The trigram analyzer in sentence mode with normal sensitivity returned the best results across the board for determining joint sentiment. For independent positive/negative sentiment, the results are less clear. In particular, the trigram analyzer was not good at detecting negative sentiment unless the sensitivity was increased, which somewhat reduced accuracy for joint sentiment.

8.1.3 Phrasal

The phrasal algorithm did not excel in any area. It was roughly equivalent to the bag-of-words in non-sentence mode for joint and positive sentiment, but suffered from poor recall with negative sentiment (though not as badly as the various trigram configurations). Despite these results, however, I believe the phrasal algorithm deserves further development.

I also ran the phrasal algorithm with the polarity-reversal-detection feature disabled (the last row of the table), and it's worth noting that the reversal feature did improve accuracy somewhat. Review of the annotator output confirmed that some reversals were found; document 2914 was one where the reversed sentiment was correct (though this could arguably be considered an artifact, based on the relationship between the sentiment-bearing term and the reversal term that was identified).

8.2 Algorithm Performance

One issue is the run-time performance of the analyzers: CPU, memory, and storage resources; time to initialize the system (often significant, since lan-

guage models and similar resources need to be loaded, configuration must be processed, etc); and time to process each review document.

While the UIMA framework is very useful and greatly reduces the time and effort required to implement these algorithms, it does add significant overhead. A typical NLP primitive operation such as part-of-speech tagging might run a couple of orders of magnitude more slowly in a UIMA implementation, all else being equal. Similarly, while OpenNLP is very powerful, it is not particularly fast at operations such as parsing.

In my testing, running these algorithms on a relatively recent and powerful PC, the bag-of-words and trigram analyzers are fast enough to be used for on-demand sentiment determination of a handful of reviews. That might be useful for doing *ad hoc* visualizations and similar exploratory research in Eli. The phrasal analyzer, on the other hand, is too slow, particularly during initialization, to be comfortably used interactively. It would be more effective for offline analysis, with results stored in the database and used later by researchers or instructors.

8.3 Significant Problems

8.3.1 Training Data

The training sets used for the machine-trained algorithms (bag-of-words and trigram) are too small, leading to sparse data issues, particularly for trigram. I believe this is a major factor in the poor results for negative-sentiment recall for the trigram algorithm in particular. Also, the training sets were created based on the numeric review ratings. That assumes that sentiment correlates to numeric ratings, which is unproven, and in fact is a hypothesis which we'd like to be able to test, but obviously cannot with a system trained under that assumption.

The experimental bigram HMM algorithm requires a large amount of human-annotated data which is laborious to produce, and if produced by a single human judge may incorporate an unacceptable degree of a single subjective interpretation of sentiment.

The phrasal algorithm builds its model using human-generated dictionaries, which avoids some of the problems with machine learning. However, it also restricts the amount of information available in the language model. It's also unknown at this point how well the word lists extracted from those dictionaries correspond to the diction employed by the student reviewers; casual inspection of the results of the algorithm (which annotates sentiment-bearing and reversal terms it finds in the documents) suggest that the lists need to be adjusted for this domain. Here the approach suggested by Gamon and Aue (Gamon and Aue, 2005), who use a minimally-supervised learning process that starts with a kernel of sentiment terms and extends it by estimating the PMI of associated terms in the training data, might be useful.

8.3.2 Sentiment Model

The representation of sentiment I used in this project, though a bit more complex than simple positive/negative, is still highly reductive. It fails to capture the wide range of sentiment that might be expressed even in a relatively narrow genre like writing peer reviews—affective responses such as "interesting" or "surprising", for example.

Also, reporting simply a single sentiment value for each document does not represent the common mix of sentiment in reviews, where the reviewer tries to describe both positive and negative aspects of the piece being reviewed.

8.3.3 Parameters

All of the algorithms (except the HMM) make use of rather arbitrary formulas with parameters that I chose intuitively or by trial and error, such as the threshold value and margin coefficient. Those parameters should be determined more rigorously, perhaps through Monte Carlo trials or, better, using an optimization approach such as Expectation Maximization.

8.3.4 Algorithm Design

The trigram algorithm almost certainly needs to implement Katz backoff (with associated discounting and smoothing) in order to produce decent probabilities.

The phrasal algorithm's use of the parse tree is primitive. After identifying a reversal term, it ought to check the tree structure to determine whether that reversal actually applies to the sentiment-bearing term in question. The phrasal annotator also really

needs a separate mode in which it builds its language model and then serializes it for fast retrieval, to reduce initialization time.

9 Further Work

9.1 Improved Sentiment Analysis

It's debatable whether it's worth further work on the bag-of-words and trigram analyzers, even though they currently return the best results. Because these algorithms do not take sentence structure and larger context into account, they will always suffer from an inability to detect non-local reversals and other subtleties.

I suggest further effort would best be put into the phrasal algorithm or other parse-tree-based approaches, and/or entirely different methods. My candidates for the latter are (probably simplified versions of) those of Nakagawa *et al.*, Greene and Resnik, and Brody and Elhadad (Greene and Resnik, 2009; Brody and Elhadad, 2010). To address the problem of non-substantive polite expressions of approval in student reviews, I believe it would be worth reviewing the method described by Taboada *et al.* (Taboada et al., 2009), which tries to identify which paragraphs and sentences in a movie review contain substantive comments. Whether this is feasible in my project, or applicable to student-writing reviews, is still an open question at this point.

Another possibly useful study is that of Andreevskaia *et al.* (Andreevskaia et al., 2007), looking at sentiment specifically in some blog genres. In some ways Eli has the "feel" of a blog, with main posts (the primary texts) and comments (the reviews), so approaches designed for blogs may be applicable; also, Eli users may tend to write in a "blog-like" fashion because the system reminds them of that environment.

9.2 Integration with Eli

Since this project was conceived as an enhancement to Eli, it should be integrated back into the Eli system at some point. That will probably take the form of a web service which will run the UIMA components and be invoked by Eli, either interactively or as a background task.

Acknowledgements

I'd like to thank Dr Chai for the suggestion which led to the project in its current formulation, for her advice and encouragement, and for an interesting and enlightening course. I'd also like to thank Bill Hart-Davidson and Mike Mcleod (WIDE Center, Michigan State University) for access to the Eli system and data, and for numerous stimulating conversations about computational rhetoric in general and student writing-review evaluation specifically; this project would not exist without them. As always, my gratitude to Malea Powell for her encouragement and support.

References

[Andreevskaia et al.2007] Alina Andreevskaia, Sabine Bergler, and Monica Urseanu. 2007. All blogs are not made equal: Exploring genre differences in sentiment tagging of blogs. In *International Conference on Weblogs and Social Media*.

[Apache Foundation2011] Apache Foundation. 2011. Apache uima. http://uima.apache.org/.

[Arora et al.2010] S. Arora, E. Mayfield, C. Penstein-Rosé, and E. Nyberg. 2010. Sentiment classification using automatically extracted subgraph features. In *Proceedings of the NAACL HLT 2010 Workshop on Computational Approaches to Analysis and Generation of Emotion in Text*, pages 131–139.

[Brody and Elhadad2010] S. Brody and N. Elhadad. 2010. An unsupervised aspect-sentiment model for online reviews. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 804–812.

[Gamon and Aue2005] Michael Gamon and Anthony Aue. 2005. Automatic identification of sentiment vocabulary: exploiting low association with known sentiment terms. In *Proceedings of the ACL 2005 Workshop on Feature Engineering for Machine Learning in NLP, ACL*, pages 57–64.

[GI] General inquirer home page. http://www.wjh.harvard.edu/~inquirer/.

[Greene and Resnik2009] S. Greene and P. Resnik. 2009. More than words: Syntactic packaging and implicit sentiment. In *Proceedings of Human Language Technologies: The 2009 Annual Conference ofthe North American Chapter of the Association for Computational Linguistics*, pages 503–511.

[Hart-Davidson et al.2010] William Hart-Davidson, Michael McLeod, Christopher Klerkx, and Michael

Wojcik. 2010. A method for measuring helpfulness in online peer review. In *Proceedings of the 28th ACM International Conference on Design of Communication - SIGDOC '10*, page 115, São Carlos, São Paulo, Brazil.

[Liu2009] Bing Liu. 2009. Sentiment analysis. *Invited talk at the 5th Annual Text Analytics Summit*.

[Nakagawa et al.2010] T. Nakagawa, K. Inui, and S. Kurohashi. 2010. Dependency tree-based sentiment classification using crfs with hidden variables. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 786–794.

[Pearson2011] Tony Pearson. 2011. How to build your own "watson jr." in your basement. https://www.ibm.com/developerworks/mydeveloperworks/blogs/InsideSystemStorage/entry/ibm_watson_how_to_build_your_own_watson_jr_in_your_basement7?lang=en, February.

[Taboada et al.2009] M. Taboada, J. Brooke, and M. Stede. 2009. Genre-based paragraph classification for sentiment analysis. In *Proceedings of the SIGDIAL 2009 Conference: The 10th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 62–70.

[Wei and Gulla2010] W. Wei and J. A Gulla. 2010. Sentiment learning on product reviews via sentiment ontology tree. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 404–413.

[WIDE2011b] WIDE. 2011b. Writing in digital environments (wide) research center. http://wide.msu.edu.

[Wiebe2002] Janyce Wiebe. 2002. MPQA releases - corpus and opinion recognition system. http://www.cs.pitt.edu/mpqa/.

[Wojcik2009] Michael Wojcik. 2009. Estimating ethos. http://www.ideoplast.org/rw/portfolio-2007-2008/ee/index.html.

[Wojcik2011] Michael Wojcik. 2011. Inventing computational rhetoric. Presentation at the 62nd annual convention of the Conference on College Composition and Communication., April.